システムソフトウェアと
オペレーティング・システム 77－41
マルチメディア通信と分散処理 87－41
（１９９８．２．２７）

「仮想ＯＳ・仮想ネットワークによる分散型組込みシステム開発環境」

吉田 利夫　茅野 眞一郎　鈴木 文雄　小泉 寿男
三菱電機株式会社

マイクロプロセッサ、リアルタイムＯＳを応用した「組込みシステム」は、専用のハードウェアが用いられ、ハードウェアとソフトウェアの開発が同時並行して行われる事が一般的である。その際、ハードウェア開発の遅れが全体の開発期間のネックになる場合が多い。また、最近は複数の機器をネットワークに接続した「分散型組込みシステム」が数多く開発されている。この様なシステムを効率良く開発するためには、専用のハードウェアや実際のネットワークを使用せずにソフトウェアの検証が可能な開発環境、即ちシミュレーションを主体とした開発環境の実現が必要である。本稿では「仮想リアルタイムＯＳ」と「仮想ネットワーク」を組み合わせた「分散型組込みシステム開発環境」の提案を行う。

A Development Method for Embedded Systems in Distributed Environment using a Virtual OS and Virtual Network

Toshio Yoshida, Shinichiro Chino, Fumio Suzuki, Toshio Koizumi
Mitsubishi Electric Corp.

Embedded Systems that utilize micro-processors and Real-time OS are usually built on dedicated hardware, and the development of software and hardware is often done concurrently. Distributed Embedded Systems that connect many embedded systems in a communication network are being developed for various applications. The unavailability of dedicated hardware and network causes delays in the software development schedule. A virtual environment is necessary for more effective software development. This paper proposes a development Virtual Real-time OS and Virtual Network based software development environment for Distributed Embedded Systems.

1. はじめに

マイクロプロセッサ、リアルタイムＯＳを応用した「組込みシステム」は、専用のハードウェアが用いられハードウェアとソフトウェアの開発が同時並行して行われる事が一般的であり、その際、ハードウェア開発の遅れが全体の開発期間のネックになる場合が多い。そのため、効率良くソフトウェアを開発するためには、専用のハードウェアを使用せずにソフトウェアの検証が可能である開発環境、即ちシミュレーションを主体とした仮想環境の実現が必要である。ワークステーションやパソコン上で組込みソフトウェアの複数のタスクを同時に実行させながらシステム検証を行うためには、組込み機器で使用されるリアルタイムＯＳのシミュレータ、即ち仮想リアルタイムＯＳが必要となる。また、最近は複数の機器をネットワークに接続した分散型組込みシステムが数多く開発されている。この場合、機器間の通信処理ソフトウェアの検証のために、仮想的なネットワークシステムが必要となる。仮想ＯＳ上の各タスクが仮想ネットワークを通じて互いに通

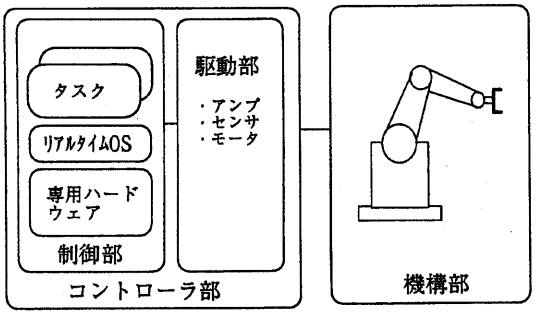

図1　組み込みシステムの構成例（ロボット）

信しあうことにより、分散型組込みシステムのシミュレーションを行い、これにより分散型組込みシステムのシステムの検証が可能となる。この様な「仮想リアルタイムＯＳ」と「仮想ネットワーク」を組み合わせた「分散型組込みシステム開発環境」により、専用のハードウェアや物理的なネットワークを使用せずにソフトウェアの検証が可能となり、組込みソフトウェア開発効率の大幅な向上が期待できる。

2. 分散型組込みシステム開発の課題

マイクロプロセッサを応用した組込みシステムの典型的な例は、ＦＡ機器である。ロボット、ＮＣ工作機械等のＦＡ機器は、一般にコントローラと機構部から構成される。さらに、コントローラは制御部と駆動部を持ち、制御部はマイクロプロセッサを搭載した専用ハードウェアから、駆動部はアクチュエータとセンサ、アクチュエータの駆動のための駆動アンプ等から構成される。図1は産業用ロボットの構成の例を示している。ロボットはコントローラと機構部から構成され、コントローラは制御部と駆動部からなり、駆動部はサーボモータ及びその駆動アンプ、モータの回転位置のセンサなどから構成される。制御部の専用ハードウェア上には、リアルタイムＯＳ、デバイスドライバ、タスク等の、ロボットの運動制御や他の機器との通信制御等のための組込みソフトウェアが搭載される。組込みシステムでは、機構部ばかりでなくコントローラ部も専用のハードウェアが使用され、ハードウェアとソフトウェア開発が同時に並行して行われる事が一般的である。そのためソフトウェアデバッグの開始がハードウェアの完成まで待たされたり、開発初期の動作が不安定なハードウェアの使用を余儀なくされたりして、効率の良いソフトウェア検証が出来ない場合が多い。またシステムが出荷された後は稼働中のシステムを長時間にわたって停止させることが難しく、実際のハードウェアを使用したソフトウェアのテストが十分出来ない。このようなハードウェアネックを回避し、効率良く組込みソフトウェアを開発するためには、専用のハードウェアを使用しない組込みソフトウェアの開発環境、即ちシミュレーションを主体とした仮想環境の実現が必要である。

また、最近は複数の機器をネットワークに接続した分散型組込みシステムが数多く開発されている。例えば、工場の製造ラインには、ロボット、シーケンサ、ＮＣ工作機械等の製造機器とパソコンをネットワークで接続したＦＡセルシステムが多く用いられる。図2は、複数のロボットが協調して作業を行う製造ラインを制御するための、ＦＡシステムのソフトウェア構成を示している。この様な各機器をネットワークで接続した「分散型組込みシステム」については、多数の機器、パソコン、ＩＣＥなどのデバッグツールを揃えてシステムのデバッグ環境を構築する必要があり、ソフトウェアの検証を行う以前にハードウェアの準備に大きな工数が必要となる。この様な問題を解決するためには、仮想環境のなかで、各機器の挙動ばかりではなくネットワーク上の各機器間の通信のシミュレーションを行う必要がある。

組込みシステムの機器レベルでのシミュレー

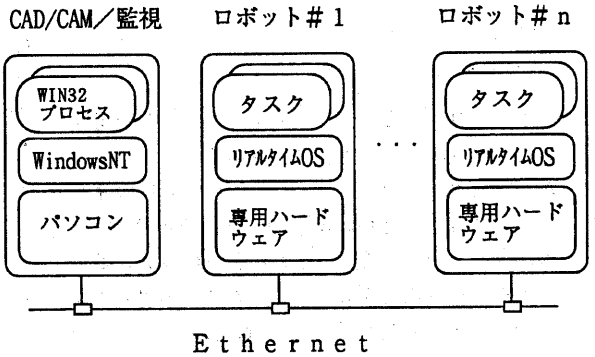

図2　分散型組込みシステムのソフトウェア構成例

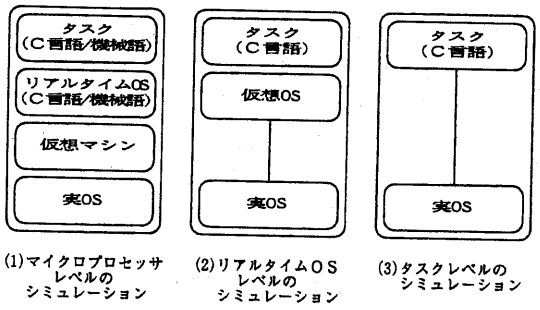

図3　組込みシステムのシミュレーションの方式

ションについては、図3に示す様に次の3方式が考えられる。

(1) マイクロプロセッサレベルのシミュレーション

各機器の組込みソフトウェアの実行命令コードを解析しながらシステムシミュレーションを行う方式である。命令の解釈だけで、バスサイクルのタイミングを考慮しない方式と、バスサイクルまで考慮したシミュレーション方式に分かれる。通常、前者で実時間の数十倍～数百倍、後者で実時間の数万倍以上のシミュレーション時間を要するため、この命令レベルのシミュレーションは、デバイスドライバ開発やＡＳＩＣ開発などの特定の目的以外には適応し難い。

(2) リアルタイムＯＳレベルのシミュレーション

UNIXやWindowsNT等の標準的な開発用ＯＳのうえにリアルタイムＯＳの仮想ＯＳを搭載し、複数のタスクを走らせながらシステムデバッグを行う。仮想ＯＳ及びタスクは開発マシンのプロセスとして実行されるので、開発マシンと専用ハードウェアの性能との関係にもよるが、実時間に対して同等以上のシミュレーション時間が期待でき、実時間よりも高速にシステムデバッグが可能である場合もある。

(3) タスクレベルのシミュレーション

いわゆるタスク単体デバッグのレベルである。前記の方式と異なりシステムデバッグが出来ないので、システム開発の仮想環境に展開できない。

このように、システムデバッグが可能であり時間的に実用に耐えるものは、ＯＳレベルのシミュレーションである。そこで、ＯＳレベルのシミュレーション、即ち仮想ＯＳを核とした組込みソフトウェア開発のための開発環境を提案する。

3. 分散組込みシステム開発方式の提案

3.1 基本方式

本稿で提案する「分散型組込みシステムの開発環境」の全体構成を図4に示す。図4の実ＯＳとは、UNIX、WindowsNTなど組込みソフトウェアの開発環境を実現するためのプラットフォームとなるＯＳである。リアルタイムノードは仮想ＯＳ（ＶＯＳ）とその上のタスクから構成され、ロボット等の各機器の組込みソフトウェアの部分に対応する。即ち図2のパソコンの部分が図4の実ＯＳノードに対応し、図2のロボット＃１～＃ｎが図4のリアルタイムノードの＃１～＃ｎにそれぞれ対応している。ネットワークに接続された「分散型組込みシステム」をシミュレーションするために、各リアルタイムノードの通信プロトコルスタックは仮想ネットワーク（ＶＮＥＴ）に接続される。仮想ネットワークは実際のネットワークの通信媒体に相当する役割を果たし、リアルタイムノード間の通信が可能となる。また、図4に示す様に、実ＯＳ側の通信プロトコルスタックもＶＮＥＴに接続する。これにより、実ＯＳ側のプロセスとＶＯＳ側のタスク間の通信もＶＮＥＴを通じて可能となり、さらに実ＯＳ側のネットワークシステムをルータとして機能させることにより、リアルタイムノードの各タスクは物理ネットワークに接続される外部の機器との通信も可能となる。現実のネットワークシステムの挙動を正確にシミュレーションするために、リアルタイムノードについても通信プロトコルのフルスタックを

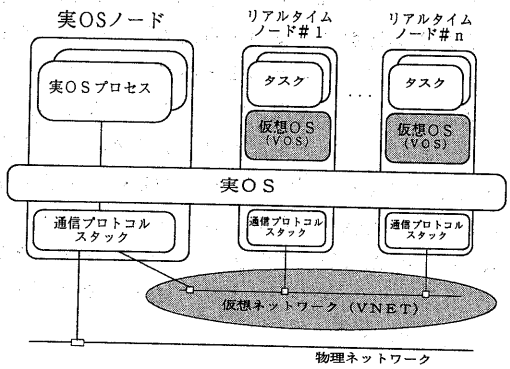

図4　仮想OS及び仮想ネットワークを核とした分散組込みシステム開発環境

実装することが必要である。これにより、例えば通信プロトコルにTCP/IPを使用した場合には、各リアルタイムノードが独立したＩＰアドレスを有しネットワーク上で独立したノードとして取り扱えるため、実際の機器の正確なシミュレーションが可能となる。

3.2 仮想リアルタイムＯＳ（ＶＯＳ）

仮想リアルタイムＯＳの実現方式には、幾つかの方式が考えられる。分散組込みシステムの開発環境として使用される仮想ＯＳは、下記の条件を満足している必要がある。

(1)実際のコントローラで使用されるリアルタイムＯＳと同一のインタフェースを持つこと

(2)シミュレーションされるタスクの挙動が現実の組込みシステムと同じであること

以上のことから、仮想リアルタイムＯＳについては実際のリアルタイムＯＳを開発のプラットフォームとなるＯＳ（実ＯＳと呼ぶ）上に移植することにより実現されるべきである。以下に仮想リアルタイムＯＳの実現方式について提案する。

仮想リアルタイムＯＳとそのタスクは、実ＯＳの一つのプロセスにマッピングする。また、仮想ＯＳ上の各タスクについては、同一プロセスのなかのスレッドにそれぞれマッピングする。実ＯＳの同一プロセス内のスレッドはメモリ空間を共有するため、仮想ＯＳ上のタスク間でメモリ空間を共有する事が出来る。このメモリ空間内に仮想ＯＳで使うシステムメモリプール及びタスクが使用するユーザメモリプールを確保する。タスクスケジューラは実際のリアルタイムＯＳと同様のタスク実行制御を実現するために、リアルタイムＯＳのロジックをそのまま移植する。但しタスクのディスパッチ処理、コンテキストスイッチなどは、実ＯＳのスレッド制御のシステムコールを使用することにより仮想ＯＳの構築が容易になる。即ち、仮想リアルタイムＯＳのスケジューラは、実ＯＳのスレッド制御のシステムコールを使用してタスクの起動停止を行うことにより、そのタスクの実行制御を行う。

3.3 仮想ネットワーク（ＶＮＥＴ）

ＶＯＳ間で共有するメモリを同軸ケーブル等のネットワークの物理的なメディアの代わりに使用して、仮想的な通信メディアとする。通信プロトコルスタックは、現実のネットワーク通信を正確にシミュレートするためにフルスタック実装する必要がある。

4. 分散型組込みシステム開発環境の実現

WindowsNT 上に仮想ＯＳ、仮想ネットワークを搭載し、分散組込みシステム開発環境の構築を行った。本節ではその構成及び実装の内容について報告を行う。

4.1 全体システム構成

本システムは、WindowsNTを実ＯＳとしたパソコンをプラットフォームとしており、図5に全体構成図を示す。WindowsNTを選択したのは、スレッド機能がサポートされているため、前節で述べたように仮想ＯＳの実現が容易なためである。図の右上に、仮想ＯＳ及びリアルタイムタスクから構成されるリアルタイムノードを示している。それぞれのリアルタイムノードは、専用のハードウェアを持つ組込みシステムに対応している。これらのリアルタイムノードは仮想ネットワーク（ＶＮＥＴ）に接続されており、仮想的な分散型組込みシステムを構成している。

また、各リアルタイムノードはWin32のプロセスともＶＮＥＴを通じて通信ができ、さらにイーサネットに繋がれた外部のパソコンや組込みシステムとの通信が可能である。外部との通信の場

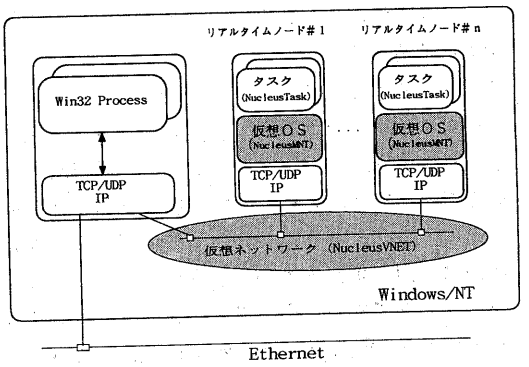

図5　分散組込みシステム開発環境の実装例

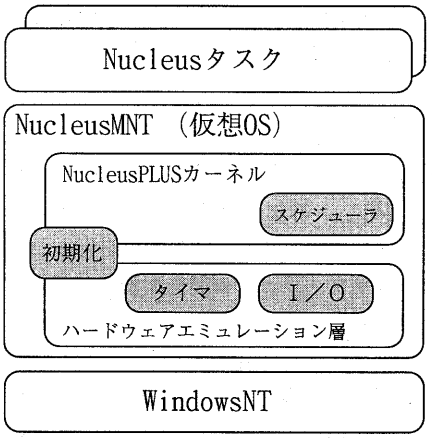

図6　仮想OS（NucleusMNT）の実装

合には、WindowsNTのネットワークシステムがルータとしての役割を担当している。これにより、パソコンの性能が不足する場合に複数台のWindowsNT機をイーサネットに接続した仮想環境を実現する事が出来る。

4.2 仮想リアルタイムＯＳ（ＶＯＳ）の実装

米国ＡＴＩ社と共同で、市販のリアルタイムＯＳであるNucleusPLUSをベースに、WindowsNT上の仮想ＯＳ（ＶＯＳ）としてNucleusMNTを開発した。実際のコントローラで使用されている環境をシミュレーションすることを可能とするため、カーネルとして市販リアルタイムＯＳを用いた。NucleusMNTは、Nucleus PLUSのエミュレーションのために、NucleusPLUSに対して次のような変更を加えている。（図6参照）

- 初期化モジュールの追加、変更
- ハードウェアエミュレーション層の追加
- スケジューラの変更

初期化モジュールに関してはタイマ、Ｉ／Ｏの割込みベクトルの変更等を行い、ハードウェアエミュレーション層としてタイマ、Ｉ／Ｏのエミュレーションを追加した。スケジューラについては、NucleusPLUSのタスク実行制御部分をWindowsNTのスレッド制御機能を利用する方式に変更した。

4.3 仮想ネットワーク（ＶＮＥＴ）の実装

TCP/IPプロトコルをサポートするＡＴＩ社のNucleusNETをベースに、WindowsNT上に仮想ネットワークを開発した。図7は仮想ネットワーク（ＶＮＥＴ）の全体構成を示しており、ここでは仮想ネットワークを構成する各モジュールの説明を行う。

(1) ＮｕｃｌｅｕｓＭＮＴ

市販のリアルタイムＯＳであるNucleusPLUSをベースとした仮想ＯＳである。

(2) プロトコルスタック

ＡＴＩ社のNucleusNETをＩＰのプロトコルスタックとして利用している。各リアルタイムノードはそれぞれ独立したプロトコルスタックを持つ。

(3) ＶＮＥＴＤＲＶ

VNETDRVは仮想ＯＳの仮想ネットワークに対するドライバであり、WindowsNTのデバイスドライバとして実装されている。

(4) 共有メモリ

仮想ネットワークでは、リアルタイムノード間の共有メモリを仮想のネットワークメディアとして使用する。この共有メモリは、次の4つの領域に分けて使用している。

- 共有メモリ管理領域
- パケット管理領域
- パケットキュー領域

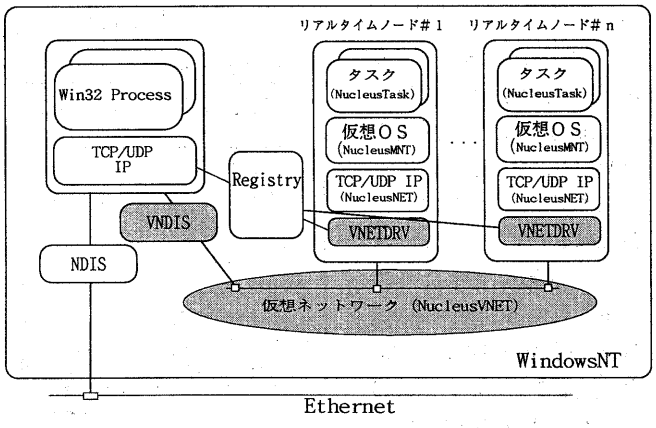

図7　仮想ネットワーク NucleusVNET の実装

• パケットデータ領域

(5) ＶＮＤＩＳ

VNDISドライバは、仮想ネットワークに対するWindowsNTのネットワークドライバである。本ドライバにより、WindowsNTのプロセスとＶＮＥＴ上の各ノードのタスクの間での通信が可能となる。さらにＩＰルーティング機能により、ＶＮＥＴ上の各ノードはイーサネットを介して他のコンピュータ、機器と通信可能である。

(6) ＲＥＧＩＳＴＲＹ

仮想ネットワークの構成管理情報を、WindowsNTのVNETDRV関連レジストリ情報として持っている。構成管理データとしては、各ノードのインターネットアドレス、イーサネットアドレス等があり、ユーティリティソフトウエアを使用して設定、変更を行う。

5. 仮想ネットワークの性能評価

ＶＮＥＴを利用した通信構成(図8参照)における性能計測結果を表１に示す。この計測では、異なるノードあるいは仮想ノード間において、少量データ(48Byte)の転送とそれに対する応答を行うクライアントとサーバのタスクを実行し、応答時間を計測した。実験に使用したシステムは、CPUがPentium-166MHz、メモリは64MBのシステムである。

表１より、NucleusMNT間での通信がWindowsNTのプロセスとの通信よりも遅いことが判る。WindowsNTの通信系はＯＳ内のシステムサービスとして実装されており、スケジューリングの影響を受けないのに対して、NucleusVNETの通信系はNucleusMNT上のタスクとして実装されており、WindowsNTのスケジューリングの影響を受けるためと考えられる。通信系の実装方法を変えることにより、実際のリアルタイムＯＳでの挙動と相違がでるが、ＶＮＥＴのパフォーマンスを上げることは可能と考えられる。

6. 終わりに

仮想ＯＳ(ＶＯＳ)、仮想ネットワーク(ＶＮＥＴ)を開発し、これらを核にした「分散組込みシステム開発環境」についての報告を行った。本システムはソフトウェアの開発環境であるが、パソコンを使用した組込みシステムのラピドプロトタイピングツールと考えられる。さらに駆動部、機構部のシミュレーションを充実させることにより、開発環境に止まらず各種の分散型組込みシステムのシミュレーションシステムとしても展開出来る。

参考文献

[1] J.A.Rowson, Hardware/Software Co-Simulation, Proceedings of 31st Design Automation Conference,1994

[2] Nucleus PLUS Target Specific Notes SunOS, ATI 社

[3] VxSim Release 5.3 User's Guide, Wind River Systems 社

表１　ＶＮＥＴの性能計測結果 (ms)

		ﾃｽﾄ1	ﾃｽﾄ2	ﾃｽﾄ3	ﾃｽﾄ4	参考ﾃｽﾄ
TCP	MIN	8.4	6.1	7.3	10.9	4.0
	TYP	8.5	6.1	7.5	11.1	4.1
	MAX	10.6	7.4	12.3	12.7	5.1
UDP	MIN	9.0	5.6	6.3	10.6	1.7
	TYP	9.1	5.6	6.4	10.7	1.7
	MAX	9.8	6.8	6.7	11.5	2.3

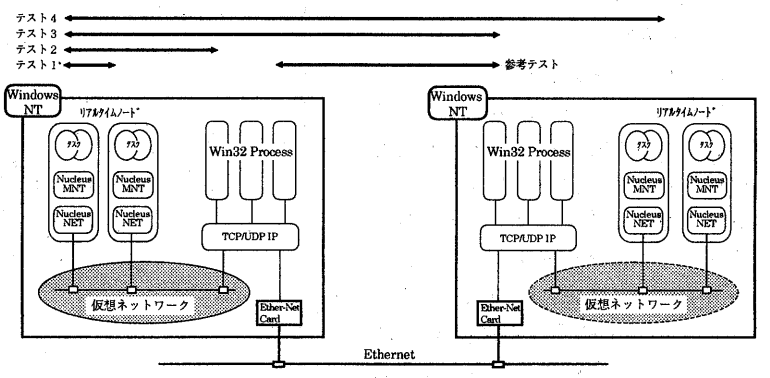

図8　ＶＮＥＴの性能計測